

05

# 데이터베이스 프로그래밍

# 목차

**01**

데이터베이스 프로그래밍의 개념

**02**

저장 프로그램

**03**

데이터베이스 연동 자바 프로그래밍

**04**

데이터베이스 연동 웹 프로그래밍

# 학습목표

- ❖ 데이터베이스 프로그래밍의 개념을 이해한다.
- ❖ 저장 프로그램의 문법과 사용 방법을 알아본다.
- ❖ 자바 프로그램과 데이터베이스를 연동하는 방법을 알아본다.
- ❖ JSP 프로그램과 데이터베이스를 연동하는 방법을 알아본다.

## 01 데이터베이스 프로그래밍의 개념



# 데이터베이스 프로그래밍의 개념

## ❖ 프로그래밍이란?

- 프로그램을 설계하고 소스코드를 작성하여 디버깅하는 과정

## ❖ 데이터베이스 프로그래밍이란?

- DBMS에 데이터를 정의하고 저장된 데이터를 읽어와 데이터를 변경하는 프로그램을 작성하는 과정
- 일반 프로그래밍과는 데이터베이스 언어인 SQL을 포함한다는 점이 다름

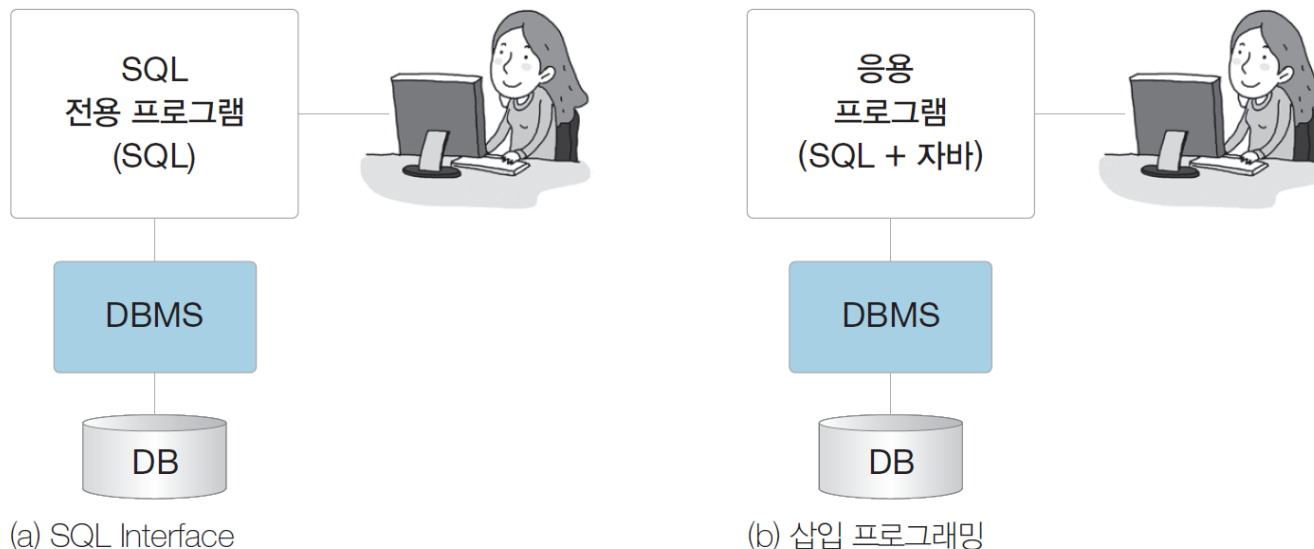


그림 5-1 데이터베이스 프로그래밍

# 데이터베이스 프로그래밍의 개념

## ❖ 데이터베이스 프로그래밍 방법

### ❶ SQL 전용 언어를 사용하는 방법

SQL 자체의 기능을 확장하여 변수, 제어, 입출력 등의 기능을 추가한 새로운 언어를 사용하는 방법.  
오라클은은 저장 프로그램 언어를 사용하며, SQL Server는 T-SQL이라는 언어를 사용함.

### ❷ 일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법

자바, C, C++ 등 일반 프로그래밍 언어에 SQL 삽입하여 사용하는 방법.

일반 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리, 검색함.  
삽입된 SQL문은 DBMS의 컴파일러가 처리함.

### ❸ 웹 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법

호스트 언어가 JSP, ASP, PHP 등 웹 스크립트 언어인 경우다.

### ❹ 4GL(4th Generation Language)

데이터베이스 관리 기능과 비주얼 프로그래밍 기능을 갖춘 'GUI 기반 소프트웨어 개발 도구'를 사용하여 프로그래밍하는 방법. Delphi, Power Builder, Visual Basic 등이 있음.

# 데이터베이스 프로그래밍의 개념

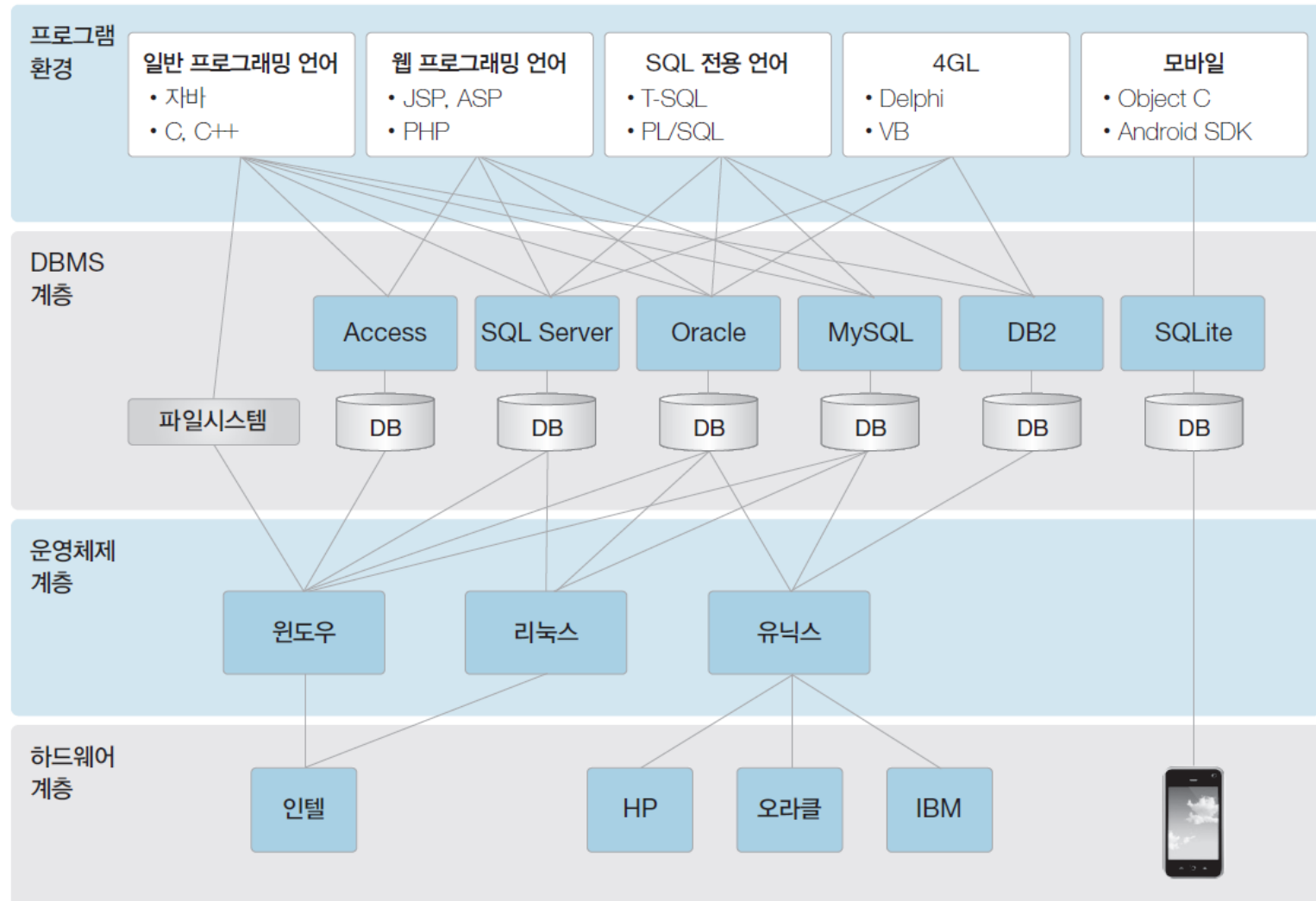


그림 5-2 DBMS 플랫폼과 데이터베이스 프로그래밍의 유형

# 데이터베이스 프로그래밍의 개념

표 5-1 DBMS의 종류와 특징

특징	Access	SQL Server	Oracle	MySQL	DB2	SQLite
제조사	마이크로소프트사	마이크로소프트사	오라클사	오라클사	IBM사	리처드 힙 (오픈소스)
운영체제 기반	윈도우	윈도우 리눅스	윈도우, 유닉스, 리눅스	윈도우, 유닉스, 리눅스	유닉스	모바일 OS (안드로이드, iOS 등)
용도	개인용 DBMS	윈도우 기반 기업용 DBMS	대용량 데이터 베이스를 위한 응용	소용량 데이터 베이스를 위한 응용	대용량 데이터 베이스를 위한 응용	모바일 전용 데이터베이스



## 02. 저장 프로그램

1. 저장 프로그램
2. 트리거
3. 사용자 정의 함수
4. 저장 프로그램 문법 요약



# 1. 저장 프로그램

- 저장 프로그램(Stored Program) : 데이터베이스 응용 프로그램을 작성하는 데 사용하는 MySQL의 SQL 전용 언어
- SQL 문에 변수, 제어, 입출력 등의 프로그래밍 기능을 추가하여 SQL 만으로 처리하기 어려운 문제를 해결함
- 저장 프로그램은 Workbench에서 바로 작성하고 컴파일한 후 결과를 실행함

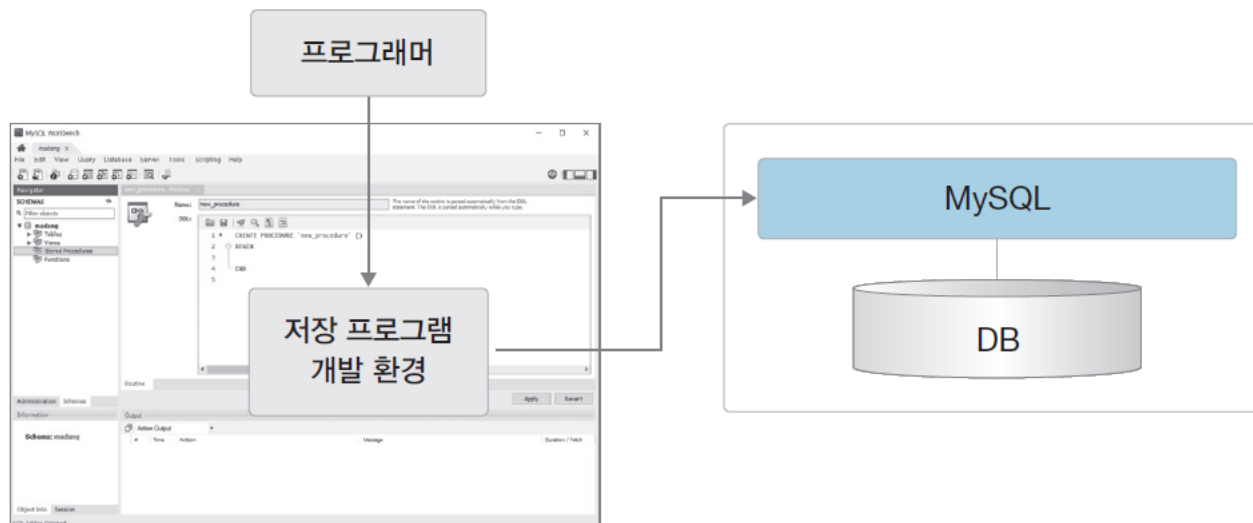


그림 5-3 저장 프로그램 개발 환경

# 1. 저장 프로그램

## ❖ 프로시저

The screenshot displays the MySQL Workbench interface with the 'dorepeat' stored procedure being defined and executed. The SQL editor contains the following code:

```
1 delimiter //
2 • CREATE PROCEDURE dorepeat(p1 INT)
3 BEGIN
4     SET @x = 0;
5     REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
6 END //
7 delimiter ;
8
9 • call dorepeat(1000);
10
11
```

Annotations on the image indicate the steps:

- ① 프로그램 정의 (Program Definition): Points to the SQL code in the editor.
- ② 실행 (Execution): Points to the 'Execute' button in the toolbar.
- ③ 실행 결과 (Execution Result): Points to the 'Output' tab showing the execution log.
- ④ 개체 확인 (Object Check): Points to the 'SCHEMAS' pane on the left, where 'dorepeat' is listed under 'Stored Procedures'.

The 'Output' tab shows the following execution results:

#	Time	Action	Message	Duration / Fetch
✓ 1	14:24:37	CREATE PROCEDURE dorepeat(p1 INT) BEGIN SET @x = 0; REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;	0 row(s) affected	0.016 sec
✓ 2	14:24:37	call dorepeat(1000)	0 row(s) affected	0.000 sec

At the bottom of the window, it says 'Query Completed'.

그림 5-4 프로시저를 정의하는 과정

# 1. 저장 프로그램

## ❖ 프로시저

### ■ 프로시저를 정의하려면 CREATE PROCEDURE 문을 사용함

### ■ 정의 방법

- 프로시저는 선언부와 실행부(BEGIN-END)로 구성됨
- 선언부에서는 변수와 매개변수를 선언하고 실행부에서는 프로그램 로직을 구현함
- 매개변수(parameter)는 저장 프로시저가 호출될 때 그 프로시저에 전달되는 값
- 변수(variable)는 저장 프로시저나 트리거 내에서 사용되는 값
- 소스코드에 대한 설명문은 /\*와 \*/ 사이에 기술
- 설명문이 한 줄이면 이중 대시(-- ) 기호 다음에 기술해도 됨

```
CREATE PROCEDURE 저장프로시저_이름(IN 매개변수, OUT 매개변수)
BEGIN
    SQL문장들
END
```

# 1. 저장 프로그램

## ❖ 프로시저

- 데이터베이스에서 처리해야 하는 어떤 논리적 순서를 구성하고 그것을 하나의 명령어로 처리할 수 있게 한 것이며, 복잡한 처리 및 조회 등의 쿼리를 작성할 때 사용
- 특징
  - SQL Server의 성능을 향상 시킬 수 있다.
  - 유지 관리가 간편하다.
  - 모듈식 프로그래밍이 가능하다.
  - 보안을 강화할 수 있다.

# 1. 저장 프로그램

## ❖ 삽입 작업을 하는 프로시저

- 프로시저로 데이터 삽입 작업을 하면 좀 더 복잡한 조건의 삽입 작업을 인자 값만 바꾸어 수행할 수도 있고, 저장해두었다가 필요할 때마다 호출하여 사용할 수도 있음

예제 5-1 Book 테이블에 한 개의 튜플을 삽입하는 프로시저

InsertBook.sql

```
01 use madang;
02 delimiter //
03 CREATE PROCEDURE InsertBook(
04     IN myBookID INTEGER,
05     IN myBookName VARCHAR(40),
06     IN myPublisher VARCHAR(40),
07     IN myPrice INTEGER)
08 BEGIN
09     INSERT INTO Book(bookid, bookname, publisher, price)
10         VALUES(myBookID, myBookName, myPublisher, myPrice);
11 END;
12 //
13 delimiter ;

A /* 프로시저 InsertBook을 테스트하는 부분 */
B CALL InsertBook(13, '스포츠과학', '마당과학서적', 25000);
C SELECT * FROM Book;
```

# 1. 저장 프로그램

## ❖ 삽입 작업을 하는 프로시저

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000

그림 5-5 InsertBook 프로시저를 실행한 후 Book 테이블

# 1. 저장 프로그램

## ❖ 제어문을 사용하는 프로시저

- 저장 프로그램의 제어문은 어떤 조건에서 어떤 코드가 실행되어야 하는지를 제어하기 위한 문법으로, 절차적 언어의 구성요소를 포함함

표 5-2 저장 프로그램의 제어문

구문	의미	문법
DELIMITER	<ul style="list-style-type: none"><li>구문 종료 기호 설정</li></ul>	DELIMITER {기호}
BEGIN-END	<ul style="list-style-type: none"><li>프로그램 문을 블록화시킴</li><li>중첩 가능</li></ul>	BEGIN {SQL 문} END
IF-ELSE	<ul style="list-style-type: none"><li>조건을 검사 결과에 따라 문장을 선택적으로 수행</li></ul>	IF <조건> THEN {SQL 문} [ELSE {SQL 문}] END IF;
LOOP	<ul style="list-style-type: none"><li>LEAVE 문을 만나기 전까지 LOOP을 반복</li></ul>	[label:] LOOP {SQL 문   LEAVE [label]} END LOOP
WHILE	<ul style="list-style-type: none"><li>조건이 참일 경우 WHILE 문의 블록을 실행</li></ul>	WHILE <조건> DO {SQL 문   BREAK   CONTINUE} END WHILE
REPEAT	<ul style="list-style-type: none"><li>조건이 참일 경우 REPEAT 문의 블록을 실행</li></ul>	[label:] REPEAT {SQL 문   BREAK   CONTINUE} UNTIL <조건> END REPEAT [label:]
RETURN	<ul style="list-style-type: none"><li>프로시저를 종료</li><li>상태값을 반환 가능</li></ul>	RETURN [<식>]



# 1. 저장 프로그램

예제 5-2 동일한 도서가 있는지 점검한 후 삽입하는 프로시저

**BookInsertOrUpdate.sql**

```
01 use madang
02 delimiter //
03 CREATE PROCEDURE BookInsertOrUpdate(
04     myBookID INTEGER,
05     myBookName VARCHAR(40),
06     myPublisher VARCHAR(40),
07     myPrice INT)
08 BEGIN
09     DECLARE mycount INTEGER;
10     SELECT count(*) INTO mycount FROM Book
11         WHERE bookname LIKE myBookName;
12     IF mycount!=0 THEN
13         SET SQL_SAFE_UPDATES=0; /* DELETE, UPDATE 연산에 필요한 설정 문 */
14         UPDATE Book SET price = myPrice
15             WHERE bookname LIKE myBookName;
16     ELSE
17         INSERT INTO Book(bookid, bookname, publisher, price)
18             VALUES(myBookID, myBookName, myPublisher, myPrice);
19     END IF;
20 END;
21 //
22 delimiter ;
```

- A -- BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분
- B CALL BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 25000);
- C SELECT \* FROM Book; -- 15번 튜플 삽입 결과 확인
- D -- BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분
- E CALL BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 20000);
- F SELECT \* FROM Book; -- 15번 튜플 가격 변경 확인

# 1. 저장 프로그램

## ❖ 제어문을 사용하는 프로시저

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000
15	스포츠 즐거움	마당과학...	25000

B행 호출 결과



bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000
15	스포츠 즐거움	마당과학...	20000

E행 호출 결과

그림 5-6 BookInsertOrUpdate 프로시저를 실행한 후 Book 테이블

# 1. 저장 프로그램

## ❖ 결과를 반환하는 프로시저

예제 5-3 Book 테이블에 저장된 도서의 평균가격을 반환하는 프로시저

AveragePrice.sql

```
01 delimiter //
02 CREATE PROCEDURE AveragePrice(OUT AverageVal INTEGER)
03 BEGIN
04 SELECT AVG(price) INTO AverageVal
05 FROM Book WHERE price IS NOT NULL;
06 END;
07 //
08 delimiter ;
09
```

```
A /* 프로시저 AveragePrice를 테스트하는 부분 */
B CALL AveragePrice(@myValue);
C SELECT @myValue;
```

@myValue
----------

15792
-------

그림 5-7 AveragePrice 프로시저를 실행한 결과

# 실습

- <학과> 테이블에 새로운 레코드를 삽입하고 삽입한 레코드를 보여주는 '새학과' stored procedure를 만드시오. 아래 실행 결과는 이 프로시저를 이용하여 출력한 예이다.

```
CALL 새학과('08', '컴퓨터보안학과', '022-200-7000');
```

실행결과	학과번호	학과명	전화번호
	08	컴퓨터보안학과	022-200-7000

- "수강신청" 데이터베이스의 총 학생 수, 교수 수, 과목 수를 계산하는 "통계" stored procedure를 만드시오. 아래 실행 결과는 이 프로시저를 이용하여 출력한 예이다.

```
CALL 통계(@a, @b, @c);  
SELECT @a AS 학생수, @b AS 교수수, @c AS 과목수;
```

실행결과	학생수	교수수	과목수
	10	7	10

# 실습

- <수강신청내역> 테이블에서 과목별로 수강자 수를 반환하는 저장 프로시저를 작성하시오.

```
CALL 과목수강자수('K20002',@Count);  
select @Count;
```

- <수강신청> 테이블에서 수강신청번호를 반환하는 저장 프로시저를 작성하시오.  
수강신청번호는 현재 이전 데이터의 다음 값을 주도록 한다

```
CALL 새수강신청('1804003', @수강신청_번호);  
SELECT @수강신청_번호;
```

# 1. 저장 프로그램

## ❖ 커서를 사용하는 프로시저

- 커서(cursor)는 실행 결과 테이블을 한 번에 한 행씩 처리하기 위하여 테이블의 행을 순서대로 가리키는 데 사용함

표 5-3 커서와 관련된 키워드

키워드	역할
CURSOR <cursor 이름> IS <커서 정의>	커서를 생성
OPEN <cursor 이름>	커서의 사용을 시작
FETCH <cursor 이름> INTO <변수>	행 데이터를 가져옴
CLOSE <cursor 이름>	커서의 사용을 끝냄

# 1. 저장 프로그램

## ❖ 커서를 사용하는 프로시저

### ■ 커서의 작동 순서

- ① 커서 선언(DECLARE)
- ② 반복 조건 선언(DECLARE, HANDLER)
- ③ 커서 열기(OPEN)
- ④ 커서에서 데이터 가져오기(FETCH)
- ⑤ 데이터 처리
- ⑥ 커서 닫기(CLOSE)

# 1. 저장 프로그램

예제 5-4 Orders 테이블의 판매 도서에 대한 이익을 계산하는 프로시저

Interest.sql

```
01 delimiter //
02 CREATE PROCEDURE Interest()
03 BEGIN
04     DECLARE myInterest INTEGER DEFAULT 0.0;
05     DECLARE Price INTEGER;
06     DECLARE endOfRow BOOLEAN DEFAULT FALSE; /* 행의 끝 여부 */
07     DECLARE InterestCursor CURSOR FOR      /* 커서 선언 */
08         SELECT saleprice FROM Orders;
09     DECLARE CONTINUE handler              /* 행의 끝일 때 Handler 정의 */
10         FOR NOT FOUND SET endOfRow=TRUE;
11     OPEN InterestCursor; /* 커서 열기 */
12     cursor_loop: LOOP
13         FETCH InterestCursor INTO Price;
14         IF endOfRow THEN LEAVE cursor_loop;
15         END IF;
16         IF Price >= 30000 THEN
17             SET myInterest = myInterest + Price * 0.1;
18         ELSE
19             SET myInterest = myInterest + Price * 0.05;
20         END IF;
21     END LOOP cursor_loop;
22     CLOSE InterestCursor;
23     SELECT CONCAT(' 전체 이익 금액 = ', myInterest);
24 END;
25 //
26 delimiter ;

A /* Interest 프로시저를 실행하여 판매된 도서에 대한 이익금을 계산 */
B CALL Interest();
```



# 1. 저장 프로그램

## ❖ 커서를 사용하는 프로시저

CONCAT(' 전체 이익 금액 = ', myInterest)
전체 이익 금액 = 5900

그림 5-8 Interest 프로시저를 실행한 결과

## 2. 트리거

- 트리거(trigger) : 데이터의 변경(INSERT, DELETE, UPDATE) 문이 실행될 때 자동으로 따라서 실행되는 프로시저

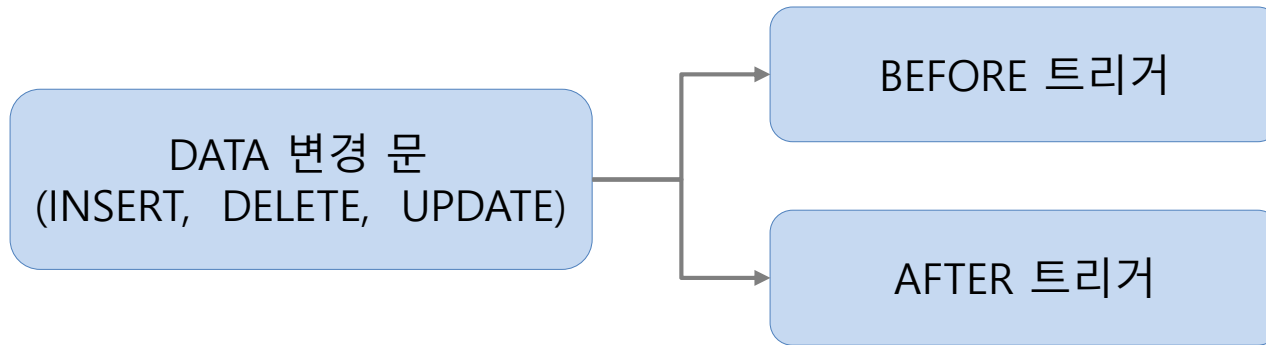


그림 5-9 데이터 변경과 트리거의 수행

## 2. 트리거

### 예제 5-5 새로운 도서를 삽입한 후 자동으로 Book\_log 테이블에 삽입한 내용을 기록하는 트리거

```
SET global log_bin_trust_function_creators=ON; /* 실습을 위해 root 계정에서 실행
```

```
/* madang 계정에서 실습을 위한 Book_log 테이블 생성해준다. */  
CREATE TABLE Book_log(  
    bookid_I INTEGER,  
    bookname_I VARCHAR(40),  
    publisher_I VARCHAR(40),  
    price_I INTEGER);
```

```
01 delimiter //  
02 CREATE TRIGGER AfterInsertBook  
03 AFTER INSERT ON Book FOR EACH ROW  
04 BEGIN  
05 DECLARE average INTEGER;  
06 INSERT INTO Book_log  
07 VALUES(new.bookid, new.bookname, new.publisher, new.price);  
08 END;  
09 //  
10 delimiter ;
```

- A /\* 삽입한 내용을 기록하는 트리거 확인 \*/
- B INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000);
- C SELECT \* FROM Book WHERE BOOKID=14;
- D SELECT \* FROM Book\_log WHERE BOOKID\_L='14' ; -- 결과 확인

## 2. 트리거

Action	Message
INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25...	1 row(s) affected

Book 테이블 Insert (B 행)



bookid	bookname	publisher	price
14	스포츠 과학 1	이상미디어	25000

Book 테이블 (C 행)



bookid_1	bookname_1	publisher_1	price_1
14	스포츠 과학 1	이상미디어	25000

Book\_log 테이블 (D 행)

그림 5-10 Book 테이블에 튜플을 삽입하여 트리거가 실행된 결과

### 3. 사용자 정의 함수

- 사용자 정의 함수는 수학의 함수와 마찬가지로 입력된 값을 가공하여 결과 값을 되돌려줌

예제 5-6 판매된 도서에 대한 이익을 계산하는 함수

**fnc\_Interest.sql**

```
01 delimiter //
02 CREATE FUNCTION fnc_Interest(
03 Price INTEGER) RETURNS INT
04 BEGIN
05 DECLARE myInterest INTEGER;
06 -- 가격이 30,000원 이상이면 10%, 30,000원 미만이면 5%
07 IF Price >= 30000 THEN SET myInterest = Price * 0.1;
08 ELSE SET myInterest := Price * 0.05;
09 END IF;
10 RETURN myInterest;
11 END; //
12 delimiter ;
```

```
A /* Orders 테이블에서 각 주문에 대한 이익을 출력 */
B SELECT custid, orderid, saleprice, fnc_Interest(saleprice) interest
C FROM Orders;
```

### 3. 사용자 정의 함수

custid	orderid	saleprice	interest
1	1	6000	300
1	2	21000	1050
2	3	8000	400
3	4	6000	300
4	5	20000	1000
1	6	12000	600
4	7	13000	650
3	8	12000	600
2	9	7000	350
3	10	13000	650

그림 5-11 Orders 테이블의 건별 이익금 계산

### 3. 사용자 정의 함수

표 5-4 프로시저, 트리거, 사용자 정의 함수의 공통점과 차이점

구분	프로시저	트리거	사용자 정의 함수
공통점	저장 프로시저		
정의 방법	CREATE PROCEDURE 문	CREATE TRIGGER 문	CREATE FUNCTION 문
호출 방법	CALL 문으로 직접 호출	INSERT, DELETE, UPDATE 문이 실행될 때 자동으로 실행됨	SELECT 문에 포함
기능의 차이	SQL 문으로 할 수 없는 복 잡한 로직을 수행	기본값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행	속성 값을 가공하여 반환, SQL 문에서 직접 사용

## 4. 저장 프로그램 문법 요약

표 5-5 저장 프로그램의 기본 문법 (계속)

구분	명령어
데이터 정의어	CREATE TABLE CREATE PROCEDURE CREATE FUNCTION CREATE TRIGGER DROP
데이터 조작어	SELECT INSERT DELETE UPDATE
데이터 타입	INTEGER, VARCHAR(n), DATE
변수	DECLARE 문으로 선언 치환(SET, = 사용)



## 4. 저장 프로그램 문법 요약

표 5-5 저장 프로그램의 기본 문법 (계속)

구분	명령어
연산자	산술연산자(+, -, *, /) 비교연산자(=, <, >, >=, <=, <>) 문자열연산자(  ) 논리연산자(NOT, AND, OR)
주석	--, /* */
내장 함수	숫자 함수(ABS, CEIL, FLOOR, POWER 등) 집계 함수(AVG, COUNT, MAX, MIN, SUM) 날짜 함수(SYSDATE, DATE, DATNAME 등) 문자 함수(CHAR, LEFT, LOWER, SUBSTR 등)
제어문	BEGIN-END IF-THEN-ELSE WHILE, LOOP
데이터 제어어	GRANT REVOKE

## 4. 저장 프로그램 문법

---

### ❖ 주석

- 실행과 관계없이 단순히 프로그램 코드에 설명
- 일부분의 실행을 잠시 막기 위해 사용
- 한 줄 주석으로 '--' 기호를 사용
- 한 줄 주석으로 '#' 기호를 사용
- 여러 줄 주석으로 /\* ... \*/ 기호를 사용

## 4. 저장 프로그램 문법

### ❖ 변수

#### ■ 사용자 정의 변수 User Defined Variables

- 쿼리문 또는 Stored Program에서 공유
- 변수의 이름은 '@' 기호로 시작
- 대소문자 구분은 하지 않음
- 변수에 값을 저장할 때는 SET 문을 사용
- SELECT 쿼리문의 실행 결과에서 열의 값을 변수에 저장할 때는 ':= ' 을 사용
  - = 은 비교연산으로 동작
- SELECT 쿼리문의 조회 결과를 INTO 절을 사용하여 변수에 대입

```
SELECT memid, memname INTO @id, @nm  
FROM members  
WHERE memid = 'kim1';
```

```
SELECT @id, @nm;
```

## 4. 저장 프로그램 문법

### ❖ 지역변수

#### ■ 지역변수를 정의하는 DECLARE 문

- DECLARE var\_name [, var\_name] ... type [DEFAULT value]
- DECLARE i INTEGER DEFAULT 0; # 지역변수 i 선언, 초기값 0

## 4. 저장 프로그램 문법

### ❖ SQL 프로그래밍 문

#### ■ BEGIN... END 문

- 하나 이상의 SQL 프로그램 코드를 블록으로 구성하기 위해 사용
- Stored Procedure를 작성하거나, 사용자 정의 함수 등을 만들 때 사용

#### ■ IF ... END IF 문

- 하나 이상의 SQL 프로그램 코드를 블록으로 구성하기 위해 사용
- Stored Procedure를 작성하거나, 사용자 정의 함수 등을 만들 때 사용

```
IF expression THEN
    statements;
ELSEIF elseif-expression THEN
    elseif-statements;
...
ELSE
    else-statements;
END IF;
```

## 4. 저장 프로그램 문법

### ❖ SQL 프로그래밍 문

#### ■ CASE ... END 문

- 조건에 따라 값을 선택하는 기능

```
CASE case_expression
  WHEN when_expression_1 THEN commands
  WHEN when_expression_2 THEN commands
  ...
  ELSE commands
END CASE;
```

## 4. 저장 프로그램 문법

### ❖ SQL 프로그래밍 문

#### ■ LOOP ... END LOOP 문

- 무한 반복으로 LOOP 문을 시작할 때 label 이름을 지정

```
[begin_label:] LOOP
```

```
실행할 문들
```

```
END LOOP [end_label]
```

## 4. 저장 프로그램 문법

### ❖ 절차적 SQL 프로그래밍에서 오류 처리

#### ■ DECLARE HANDLER 문

- DECLARE HANDLER 문을 사용하여 프로그램 실행 중 오류가 발생하거나 프로그램의 실행 상태에 따라 일련의 문장을 실행하도록 하여 중단없이 프로그램을 실행하도록 할 수 있다.

- DECLARE HANDLER 문의 기본 형식

```
DECLARE <유형> HANDLER  
FOR <조건값1> [, <조건값2>]...  
[BEGIN]  
처리할 문  
[END]
```

- <유형>은 "CONTINUE", "EXIT" 중 하나를 선택
- <조건> NOT FOUND : CURSOR와 관련하여 데이터 셋의 끝에 CURSOR가 위치했을 때 어떤 일이 발생하는지를 제어하기 위해 사용



## 6. 다음 프로그램을 프로시저로 작성하고 실행하시오.

- 데이터베이스는 마당서점을 이용한다.

(1) InsertBook() 프로시저를 수정하여 고객을 새로 등록하는 InsertCustomer() 프로시저를 작성하시오.

## 03. 데이터베이스 연동 자바 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 자바 프로그래밍

표 5-6 데이터베이스 연동 자바 프로그래밍 실습 환경

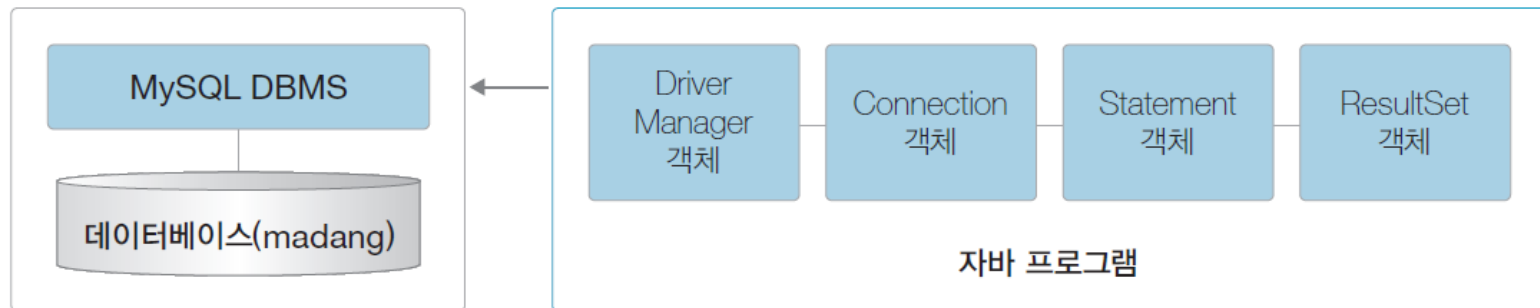
항목	프로그램
데이터베이스 프로그램	MySQL DBMS
자바 컴파일러	JDK 버전 12
데이터베이스와 자바를 연결하는 드라이버	MySQL JDBC 드라이버 Connector/J (파일명:mysql-connector-java-8.x.jar)

# 1. 소스코드 설명

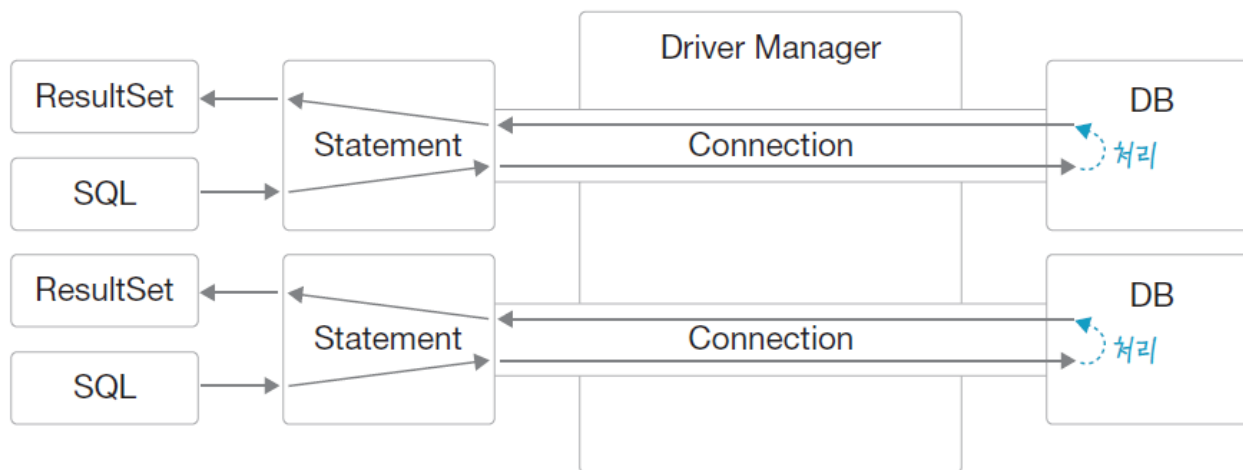
표 5-7 데이터베이스 접속 자바 클래스(java.sql)

클래스 구분	클래스 혹은 인터페이스	주요 메소드 이름	메소드 설명
java.lang	Class	Class.forName(<클래스이름>)	<클래스이름>의 JDBC 드라이버를 로딩
java.sql	DriverManager	Connection getConnection (url, user, password)	데이터베이스 Connection 객체를 생성
	Connection	Statement createStatement()	SQL 문을 실행하는 Statement 객체를 생성
		void close()	Connection 객체 연결을 종료
	Statement	ResultSet executeQuery (String sql)	SQL 문을 실행해서 ResultSet 객체를 생성
		ResultSet executeUpdate (String sql)	INSERT/DELETE/UPDATE 문을 실행 해서 ResultSet 객체를 생성
	ResultSet	boolean first()	결과 테이블에서 커서가 처음 튜플을 가리킴
		boolean next()	결과 테이블에서 커서가 다음 튜플을 가리킴
		int getInt(<int>)	<int>가 가리키는 열 값을 정수로 반환
		String getString(<int>)	<int>가 가리키는 열 값을 문자열로 반환

# 1. 소스코드 설명



(a) 자바 프로그램의 데이터베이스 연동



(b) 객체 간의 호출 순서

그림 5-12 데이터베이스 연결 자바 객체들의 호출 관계

## 2. 프로그램 실습

표 5-8 자바 프로그램 실습 단계

단계		세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정		① MySQL 8.x 설치 ② MySQL 사용자(madang)와 데이터베이스(madang) 생성	MySQL 8.x	부록 A.1~A.3 부록 B.3
[2단계] 데이터베이스 준비		① 마당서점 데이터베이스 준비 (demo_madang.sql)		부록 B.3
[3단계] 자바 실행	명령 프롬프트 이용	① 자바 컴파일러 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK JDBC	부록 C.1~C.3
	이클립스 이용	① 자바와 이클립스 개발도구 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK JDBC Eclipse	부록 C.1~C.4

## 2. 프로그램 실습

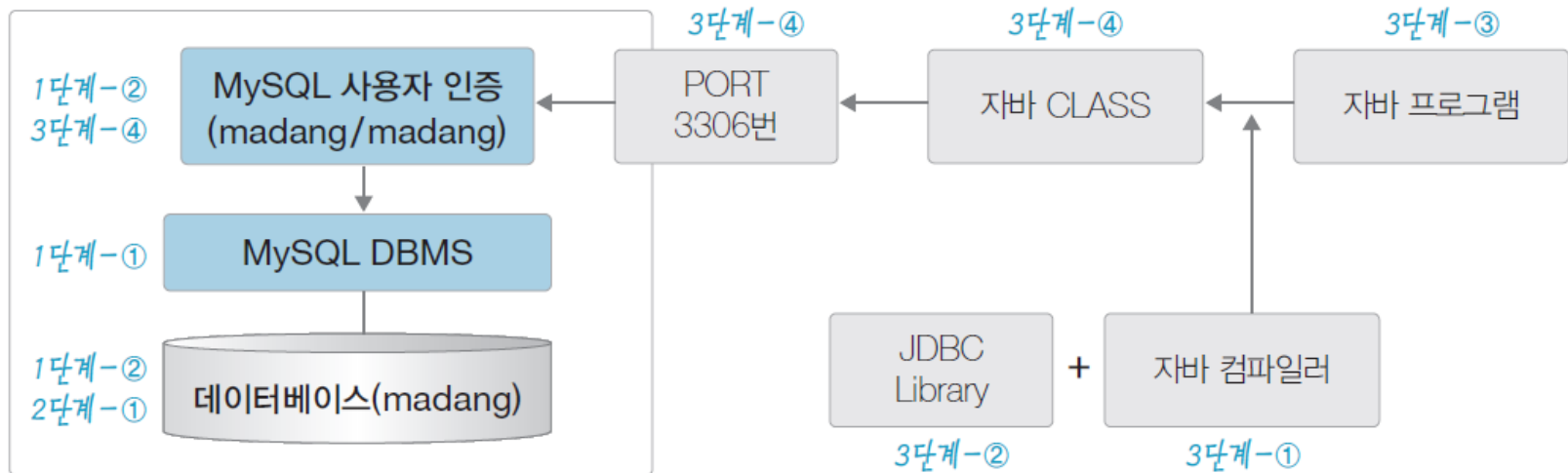


그림 5-14 데이터베이스 연동 자바 프로그램의 실행 흐름도

## 2. 프로그램 실습

### ❖ [1단계] DBMS 설치 및 환경설정

#### ① MySQL 8.x 설치

- MySQL 8.x는 이미 3장에서 설치하였다. 설치는 윈도우 10을 기준으로 한다. 아직 설치 하지 않았다면 부록 A.1~A.3을 참고하여 설치한다.

#### ② SQL 접속을 위한 사용자(madang) 설정

- MySQL에 접속하기 위한 사용자 계정(madang)과 비밀번호(madang)를 부록 B.3을 참고하여 설정한다. 정상적으로 설정되었는지 확인하기 위해 MySQL Workbench를 실행 한 후 madang/madang으로 접속해본다.



## 2. 프로그램 실습

### ❖ [2단계] 데이터베이스 준비

#### ❶ 마당서점 데이터베이스 준비(demo\_madang.sql)

- 마당서점 데이터베이스의 샘플 데이터는 이미 3장에서 설치하였다.  
이 책의 순서대로 실습을 진행하지 않았다면 부록 B.3을 참고하여 설치하면 된다.

## 2. 프로그램 실습

### ❖ [3단계(A)] 자바 실행 – 명령 프롬프트를 이용하는 방법

#### ① 자바 컴파일러 설치

- 부록 C.2를 참고하여 설치한다.

#### ② JDBC 드라이버 설치

- 부록 C.3을 참고하여 설치한다.

#### ③ 자바 프로그램 준비(booklist.java)

- booklist.java 프로그램의 소스코드는 앞에서 설명하였다. booklist.java 파일은 메모장에서 작성하거나 예제소스 폴더의 booklist.java를 가져와 사용한다.

#### ④ 컴파일 및 실행

```
C:\Documents and Settings\Myname>cd c:\W\madang
```

```
C:\madang>javac booklist.java
```

```
C:\madang>java booklist
```

## 2. 프로그램 실습

### ❖ [3단계(B)] 자바실행 – 이클립스를 이용하는 방법

#### ❶ 이클립스 개발도구 설치

- 부록 C.4를 참고하여 설치

#### ❷ JDBC 드라이버 설치

- 부록 C.3을 참고하여 설치

\* 이클립스에서 JDBC 드라이버는 프로젝트->Properties->Libraries->Add External Jars 로 간편설치

#### ❸ 자바 프로그램 준비(booklist.java)

#### ❹ 컴파일 및 실행

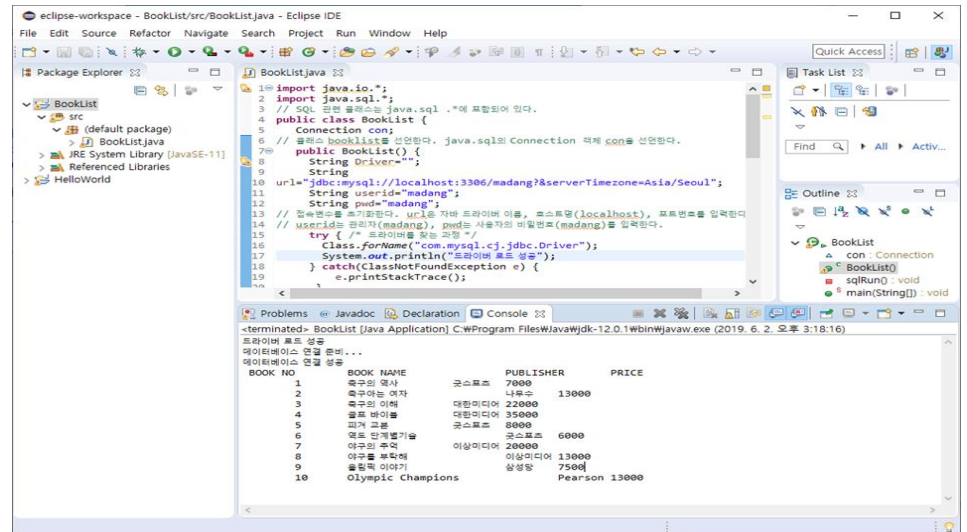


그림 5-19 이클립스에서  
booklist.java 실행 결과 화면

## 04. 데이터베이스 연동 웹 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 웹 프로그래밍

표 5-9 데이터베이스 연동 웹 프로그래밍 실습 환경

항목	프로그램
데이터베이스 프로그램	MySQL 8.x
PHP	PHP 5.x
웹서버	Apache

# 1. 소스코드 설명

- PHP 프로그램은 HTML 태그에 PHP 스크립트를 끼워 넣어 작성하는데, PHP 스크립트 부분은 `<?PHP ... ?>`에 넣어서 실행시킴.

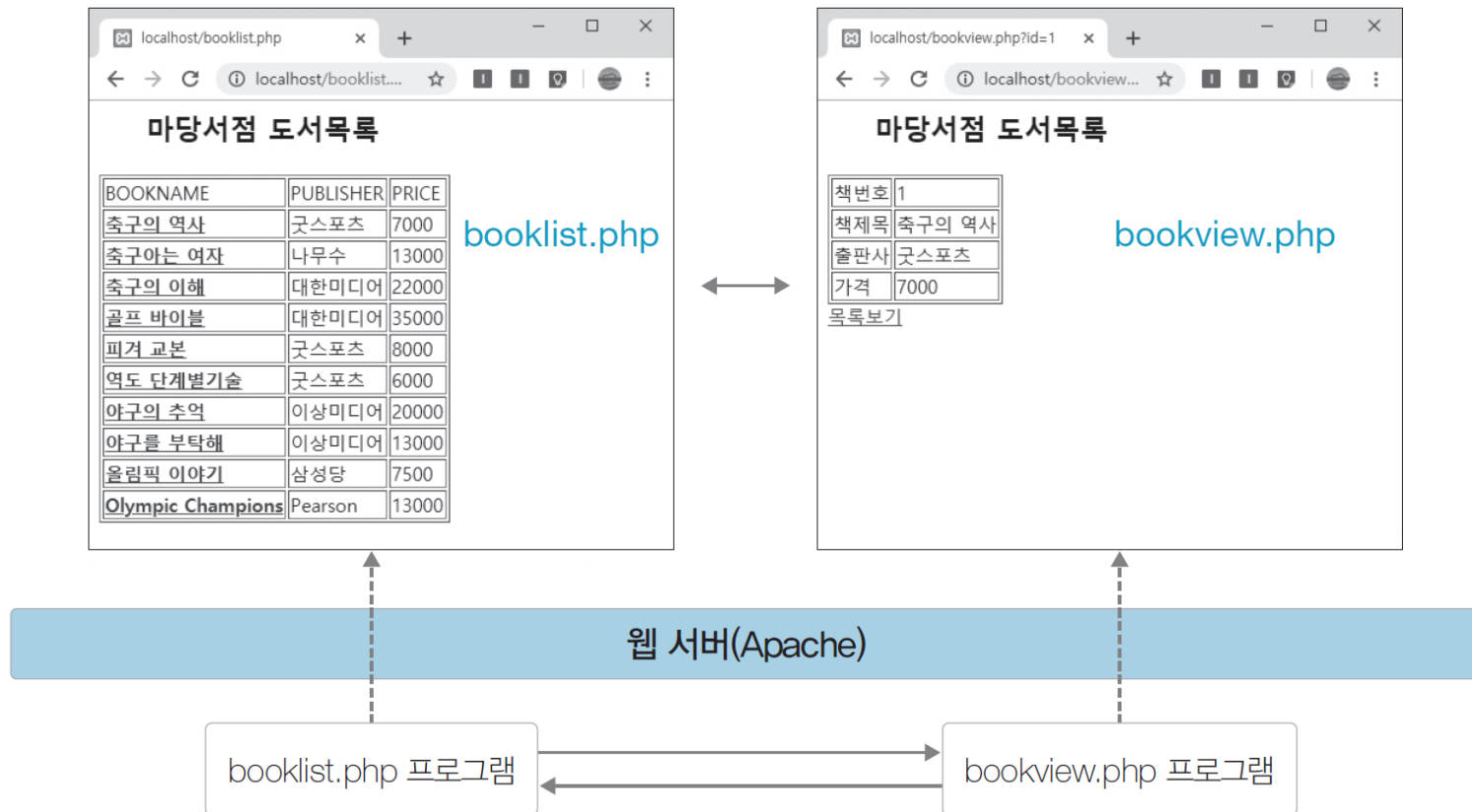


그림 5-20 `booklist.php`와 `bookview.php`의 호출 관계와 웹에서 실행된 화면

## 2. 프로그램 실습

표 5-10 JSP 프로그램 실습 단계

단계	세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정	① MySQL 8.x 설치 ② SQL 접속을 위한 사용자(madang) 및 데이터베이스(madang) 생성	MySQL 8.x	부록 A.1~A.3 부록 B.3
[2단계] 데이터베이스 준비	① 마당서점 데이터베이스 준비 (demo_madang.sql)		부록 B.3
[3단계] PHP 실행	① Apache, PHP 설치 -XAMPP 설치 (앞절에서 설명) ② PHP 프로그램 준비 (booklist.php, bookview.php) ③ 실행	XAMPP booklist.php bookview.php	부록 C.1~C.3 부록 C.5

## 2. 프로그램 실습

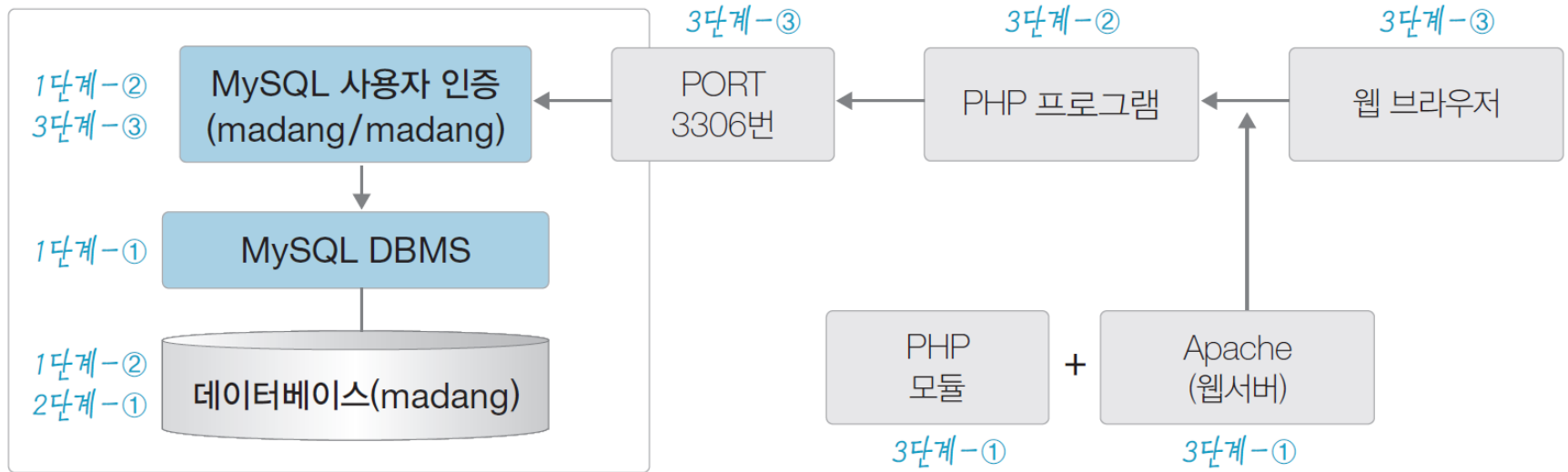


그림 5-21 데이터베이스 연동 PHP 프로그램의 실행 흐름도



## 2. 프로그램 실습

---

- [1단계] DBMS 설치 및 환경설정
- [2단계] 데이터베이스 준비
- [3단계] PHP 실행
  - ① Apache, PHP 설치 – XAMPP 설치
  - ② PHP 프로그램 준비(booklist.php, bookview.php)
  - ③ 실행

## 2. 프로그램 실습

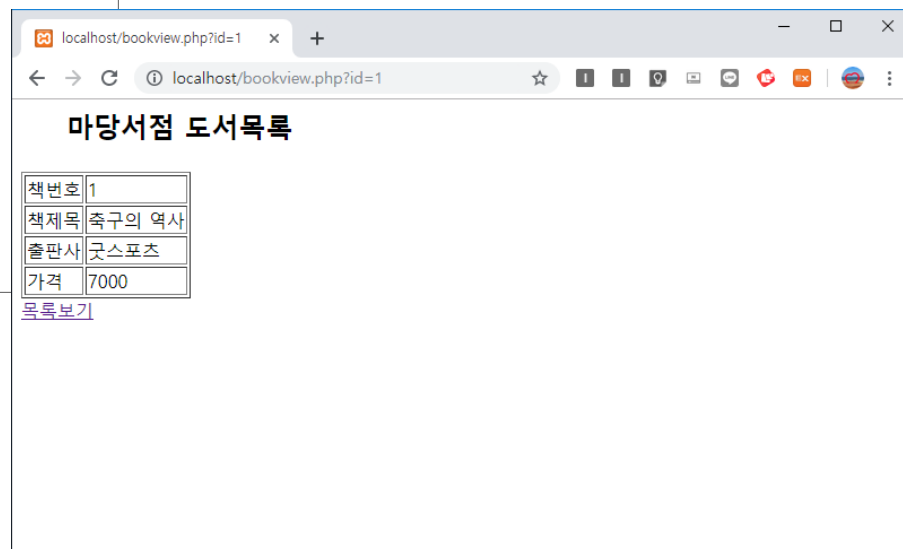
### 4 실행



마당서점 도서목록

BOOKNAME	PUBLISHER	PRICE
<a href="#">축구의 역사</a>	굿스포츠	7000
<a href="#">축구는 여자</a>	나무수	13000
<a href="#">축구의 이해</a>	대한미디어	22000
<a href="#">골프 바이블</a>	대한미디어	35000
<a href="#">피겨 교본</a>	굿스포츠	8000
<a href="#">역도 단계별기술</a>	굿스포츠	6000
<a href="#">야구의 추억</a>	이상미디어	20000
<a href="#">야구를 부탁해</a>	이상미디어	13000
<a href="#">올림픽 이야기</a>	삼성당	7500
<a href="#">Olympic Champions</a>	Pearson	13000

그림 5-23 booklist.php 실행 화면



마당서점 도서목록

책번호	1
책제목	축구의 역사
출판사	굿스포츠
가격	7000

[목록보기](#)

그림 5-24 bookview.php 실행 화면

1. 데이터베이스 프로그래밍
2. 삽입 프로그래밍
3. 저장 프로그램
4. 저장 프로시저
5. 커서
6. 트리거
7. 연동
8. JDBC(Java Database Connectivity)